The University of Texas at Austin
Department of Electrical and Computer Engineering

MIS382N: Business Data Science — Fall 2019

Problem Set Three

Caramanis/Dimakis Due: Thursday, September 26th, 2019.

Problem 1: Ridge Regression and Co-linearity.

1. Download and make sure you can run through and understand the Jupyter notebook on Ridge Regression and Colinearity we saw in class.
2. What is the test error of the "zero-variance" solution, namely, the all-zeros solution?
3. The least-squares solution does not seem to do too well, because it has so much variance. Still, it is unbiased. Show this empirically: generate many copies of the data, and for each one, obtain the least-squares solution. Average these, to show that while each run produces a beta_hat that is very different, their average begins to look more and more like the true beta.
4. Alternatively, if one had access to lots of data, instead of computing the least-square solution over smaller batches and then averaging these solutions as in the previous part of the problem, an approach is to run a single least-squares regression over all the data. Which approach do you think is better? Can you support your conclusion with experiments?

Problem 2: Starting in Kaggle.
On October 22nd, we are opening a Kaggle competition made for this class. In that one, you will be participating on your own. This is an intro to get us started, and also an excuse to work with regularization and regression which we have been discussing.

1. Let's start with our first Kaggle submission in a playground regression competition. Make an account to Kaggle and find https://www.kaggle.com/c/house-prices-advanced-regression-techniques/
2. Follow the data preprocessing steps from https://www.kaggle.com/apapiu/house-prices-advanced-regression-techniques/regularized-linear-models. Then run a ridge regression using $\alpha = 0.1$. Make a submission of this prediction, what is the RMSE you get?
(Hint: remember to exponentiate np.expm1(ypred) your predictions).
3. Compare a ridge regression and a lasso regression model. Optimize the alphas using cross validation. What is the best score you can get from a single ridge regression model and from a single lasso model?
4. Plot the l_0 norm (number of nonzeros) of the coefficients that lasso produces as you vary the strength of regularization parameter alpha.
5. Add the outputs of your models as features and train a ridge regression on all the features plus the model outputs (This is called Ensembling and Stacking). Be careful not to overfit. What score can you get? (We will be discussing ensembling more, later in the class, but you can start playing with it now).

6. (Optional)[1] Install XGBoost (Gradient Boosting) and train a gradient boosting regression. What score can you get just from a single XGB? (you will need to optimize over its parameters). We will discuss boosting and gradient boosting in more detail later. XGB is a great friend to all good Kagglers!

7. (Optional) Do your best to get the more accurate model. Try feature engineering and stacking many models. You are allowed to use any public tool in python. No non-python tools allowed.

8. (Optional) Read the Kaggle forums, tutorials and Kernels in this competition. This is an excellent way to learn. Include in your report if you find something in the forums you like, or if you made your own post or code post, especially if other Kagglers liked or used it afterwards.

9. Be sure to read and learn the rules of Kaggle! No sharing of code or data outside the Kaggle forums. Every student should have their own individual Kaggle account and teams can be formed in the Kaggle submissions with your Lab partner. This is more important for live competitions of course.

Problem 3: Optional.
Chapter 3 in the ISL book covers linear regression. The Lab at the end covers many practical aspects, including rescaling of the data, techniques for feature selection, etc. While the book is written with examples illustrated using R rather than Python, I still recommend going over the techniques and ideas described. There is nothing that conceptually is dramatically different from what we have done, however it pulls together many ideas and illustrates them in a concrete, well-written way. As I mentioned in class, you can also find the labs done in Python, if you are interested.

[1] I will assign this second half of the problem again once we discuss Boosting and Gradient boosting in a few weeks, but I include it here in case you want to give it a shot.